REMI: Mining Intuitive Referring Expressions on Knowledge Bases

Luis Galárraga, Julien Delaunay, Jean-Louis Dessalles

Short paper

REMI: Mining Intuitive Referring Expressions on Knowledge Bases

Luis Galárraga
Inria
luis.galarraga@inria.fr

Julien Delaunay
University of Rennes I
juliendelaunay35000@gmail.com

Jean-Louis Dessalles
Télécom ParisTech
dessalles@telecom-paristech.fr

ABSTRACT

A *referring expression* (RE) is a description that identifies a set of instances unambiguously. Mining REs from data finds applications in natural language generation, algorithmic journalism, and data maintenance. Since there may exist multiple REs for a given set of entities, it is common to focus on the most concise and informative (i.e., intuitive) ones. We present REMI, a method to mine intuitive REs on large knowledge bases. Our experimental evaluation shows that REMI finds REs deemed intuitive by users. Moreover we show that REMI is several orders of magnitude faster than an approach based on inductive logic programming.

1 INTRODUCTION

A *referring expression* (RE) is a description that identifies a set of entities unambiguously. For instance, the expression "x is the capital of France" is an RE for Paris, because no other city holds this title. The automatic construction of REs is a central task in natural language generation (NLG). The goal of NLG is to describe concepts in an accurate and compact manner from structured data such as a knowledge base (KB). REs also find applications in automatic data summarization, algorithmic journalism, virtual smart assistants, and KB maintenance, e.g., in query generation. Quality criteria for REs is context-dependent. For instance, NLG and data summarization aim at *intuitive*, i.e., *short* and *informative* descriptions. In this vibe, it may be more intuitive to describe Paris as "the city of the Eiffel Tower" than as "the resting place of Victor Hugo". Indeed, the world-wide prominence of the Eiffel Tower makes the first RE more informative to an average user.

Some approaches can mine intuitive REs from semantic data [1, 4–6]. Conceived at the dawn of the Semantic Web, these methods are not suitable for current KBs for three main reasons. Firstly, they cannot handle current KBs because they were designed to mine REs on scenes[1] for the sake of NLG. Scenes have much fewer predicates and instances than today's KBs. Secondly, most existing approaches are limited to conjunctive expressions on the attributes of the entities, e.g., $is(x, City) \wedge country(x, France)$. However, our experience with today's KBs suggests that this language bias does not encompass all possible intuitive expressions. For instance, to describe Johann J. Müller, we could resort to the fact that he was the supervisor of the supervisor of Albert Einstein, i.e., $supervisor(x, y) \wedge supervisor(y, Einstein)$, which goes beyond the traditional language bias due to the existentially quantified variable y. Thirdly, state-of-the-art RE miners define intuitiveness for REs in terms of number of atoms. In that spirit, the single-atom REs $capitalOf(x, France)$ and $restingPlaceOf(x, V.\ Hugo)$ are equally concise and desirable as descriptions for Paris, even though the latter may not be informative to users outside France.

[1]The exhaustive description of a place and its objects

The approach in [6] overcomes this limitation to some extent, by allowing users to provide a ranking of preference for the attributes used in the description. Nevertheless, providing such a ranking can be tedious for KBs with thousands of predicates.

We tackle the aforementioned limitations with a solution to mine intuitive REs on large KBs. How to use such REs is beyond the scope of this work, however we provide hints about potential use cases. In summary, our contributions are:

- A scheme based on information theory to quantify the intuitiveness of entity descriptions extracted from a KB.
- REMI, an algorithm to mine intuitive REs on large KBs. REMI extends the state-of-the-art language bias for REs and allows for expressions such as $mayor(x, y) \wedge party(y, Socialist)$. This design choice increases the chances of finding intuitive REs for a set of target entities.
- A user study to assess the intuitiveness of REMI's descriptions.

2 PRELIMINARIES

2.1 RDF Knowledge Bases

This work focuses on mining REs on RDF knowledge bases (KBs). A KB $\mathcal{K}$ is a set of assertions in the form of facts $p(s, o)$ with predicate $p \in \mathcal{P}$, subject $s \in \mathcal{I} \cup \mathcal{B}$, and object $o \in \mathcal{I} \cup \mathcal{L} \cup \mathcal{B}$. In this formulation, $\mathcal{I}$ is a set of entities such as London, $\mathcal{P}$ is a set of predicates, e.g., *cityIn*, $\mathcal{L}$ is a set of literal values such as strings or numbers, and $\mathcal{B}$ is a set of blank nodes, i.e., anonymous entities. An example of an RDF triple is $cityIn(London, UK)$. KBs often include assertions that state the class of an entity, e.g., $is(UK, Country)$.

2.2 Referring Expressions

2.2.1 Atoms. An *atom* $p(X, Y)$ is an expression such that p is a predicate and X, Y are either variables or constants. We say an atom has matches in a KB $\mathcal{K}$ if there exists a function $\sigma \subset \mathcal{V} \times (\mathcal{I} \cup \mathcal{L} \cup \mathcal{B})$ from the variables $\mathcal{V}$ of the atom to constants in the KB such that $\mu_\sigma(p(X, Y)) \in \mathcal{K}$. The operator μ_σ returns a new atom such that the constants in the input atom are untouched, and variables are replaced by their corresponding mappings according to σ. We call $\mu_\sigma(p(X, Y))$ a bound atom and σ a matching assignment. We extend the notion of matching assignment to conjunctions of atoms, i.e., σ is a matching assignment for $\bigwedge_{1 \le i \le n} p_i(X_i, Y_i)$ iff $\mu_\sigma(p_i(X_i, Y_i)) \in \mathcal{K}$ for $1 \le i \le n$.

2.2.2 Expressions & Language Bias. Atoms are traditionally the building blocks of referring expressions. We say that two atoms are connected if they share at least one variable argument. Most approaches for RE mining define REs as conjunctions of connected atoms with bound objects. We call this language bias, *the state-of-the-art language bias*. We extend this language by allowing atoms with additional existentially quantified variables. For this purpose, we propose subgraph expressions as the new building blocks for REs. A *subgraph expression* $\rho = p_1(x, Y_1) \wedge \bigwedge_{1 < i \le n} p_i(X_i, Y_i)$, rooted at variable x, is a conjunction of connected atoms such that for $i > 1$, atoms are

1 atom	$p_0(x, I_0)$
Path	$p_0(x, y) \wedge p_1(y, I_1)$
Path + star	$p_0(x, y) \wedge p_1(y, I_1) \wedge p_2(y, I_2)$
2 closed atoms	$p_0(x, y) \wedge p_1(x, y)$
3 closed atoms	$p_0(x, y) \wedge p_1(x, y) \wedge p_2(x, y)$

Table 1: REMI's subgraph expressions.

transitively connected to $p_1(x, Y_1)$ via at least another variable besides x. Examples are: (i) $cityIn(x, France)$, and (ii) $cityIn(x, y) \wedge officialLang(y, z) \wedge langFamily(z, Romance)$. An *expression* $e = \bigwedge_{1 \leq j \leq m} \rho_j$ is a conjunction of subgraph expressions rooted at the same variable x such that they have only x—the root variable—as common variable. Finally, we say e is a *referring expression* (RE) for a set of target entities $T \subseteq \mathcal{I}$ in a KB $\mathcal{K}$ iff:

(1) $\forall t \in T : \exists \sigma : (x \mapsto t) \in \sigma$, i.e., for every target entity t, there exists a matching assignment σ in $\mathcal{K}$ that binds the root variable x to t.
(2) $\nexists \sigma', t' : (x \mapsto t') \in \sigma' \wedge\ t' \notin T$, in other words, no matching assignment binds the root variable to entities outside the set T of target entities.

For example, consider a complete and accurate KB $\mathcal{K}$ as well as the following conjunction of two subgraph expressions:

$e = in(x, S.\ America) \wedge officialLang(x, y) \wedge langFamily(y, Germanic)$

We say that e is an RE for $T = \{Guyana, Suriname\}$ in $\mathcal{K}$ because matching assignments can only bind x to these two countries.

While we do not limit the number of subgraph expressions in REs, we do not allow more than one variable and three atoms in individual subgraph expressions, leading to the expressions in Table 1. This design decision aims at keeping both the search space and the complexity of the REs under control. Indeed, expressions with multiple non-root variables make comprehension and translation to natural language more effortful.

3 REMI

Given an RDF KB $\mathcal{K}$ and a set of target entities T, REMI returns an intuitive RE—a conjunction of subgraph expressions—that describes unambiguously the input entities T in $\mathcal{K}$. Intuitive REs are concise and resort to concepts that users are likely to understand. We first show how to quantify intuitiveness in Section 3.1. We then elaborate on REMI's algorithm in Section 3.2.

3.1 Quantifying intuitiveness

There may be multiple ways to describe a set of entities uniquely. For example, $capitalOf(x, France)$ and $birthPlaceOf(x, Voltaire)$ are both REs for Paris. Our goal is therefore to quantify the *intuitiveness* of such expressions without human intervention. We say that an RE e is more intuitive than an RE e', if $C(e) < C(e')$, where C denotes the Kolmogorov complexity. The Kolmogorov complexity $C(e)$ of a string e (e.g., an expression) is a measure of the absolute amount of information conveyed by e and is defined as the length in bits of e's shortest effective binary description [8]. If e_b denotes such binary description and M is the program that can *decode* e_b into e, $C(e) = l(e_b) + l(M)$ where $l(\cdot)$ denotes length in bits. Due to C's intractability, applications can only *approximate* it via suboptimal encodings and programs $(\hat{e_b}, \hat{M})$, hence $C(e) \approx \hat{C}(e) = l(\hat{e_b}) + l(\hat{M})$ with $C(e) \leq \hat{C}(e)$.

Our proposed encoding builds upon the observation that intuitive expressions resort to prominent concepts. For example, it is natural and informative to describe Paris as the capital of France, because the concept of capital is well understood and France is a very salient entity. In contrast, it would be more complex to describe Paris in terms of less prominent concepts, let us say, its twin cities. In this spirit, we devise a code for concepts as follows: The code for a predicate p (entity I) is the binary representation of its position k in a *ranking by prominence*. This way, prominent concepts can be rewarded with shorter codes. We can now define the estimated Kolmogorov complexity $\hat{C}$ of a single-atom subgraph expression $p(x, I)$ as:

$$\hat{C}(p(x, I)) = l(k(p)) + l(k(I \mid p))$$

In the formula, $l(\cdot) = log_2(\cdot) + 1$, $k(p)$ is p's position in the ranking of predicates of the KB, and $k(I \mid p)$ is I's conditional rank given p, i.e., I's rank among all objects of p. The latter term follows from the chain rule of the Kolmogorov complexity. For instance, if p is the predicate *city mayor*, the chain rule models the fact that once the concept of mayor has been conveyed, the context becomes narrower and the user needs to rank fewer concepts, in this example, only city mayors. The chain rule also applies to subgraph expressions with multiple atoms. For instance, the complexity of $\rho = mayor(x, y) \wedge party(y, Socialist)$ is:

$$\hat{C}(\rho) = l(k(mayor)) + l(k(party(y, z) \mid mayor(x, y))) + l(k(Socialist \mid mayor(x, y) \wedge party(y, z)))$$

The second term in the sum amounts to the code length of the rank of predicate *party* among those predicates that allow for subject-to-object joins with *mayor* in the KB. Likewise, the complexity of the Socialist party in the third term depends on the ranking of parties with mayors among their members, i.e., the bindings for z in $mayor(x, y) \wedge party(y, z)$. If a city can be unambiguously described as $mayor(x, I)$ for a non-prominent mayor I, we may achieve a shorter code length if we replace I by a variable y, an additional predicate, and a well-known party.

In line with other works that quantify prominence for concepts in KBs [7], we rank concepts by frequency (*fr*), and Wikipedia's page rank (*pr*). We denote the resulting complexity measures using these prominence metrics by $\hat{C}_{fr}$ and $\hat{C}_{pr}$ respectively.

Finally, we can estimate the Kolgomorov complexity of an RE $e = \bigwedge_{1 \leq i \leq m} \rho_i$ as the sum of the complexities of its individual subgraph expressions, i.e., $\hat{C}(e) = \sum_{1 \leq i \leq m} \hat{C}(\rho_i)$.

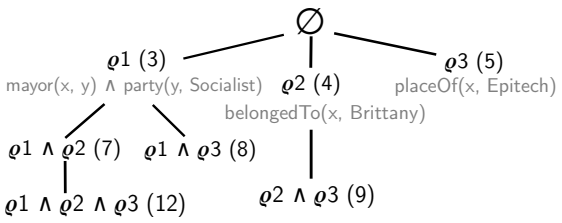

Figure 1: Search space example.

3.2 Algorithm

REMI implements a depth-first search (DFS) on conjunctions of the subgraph expressions common to **all** the target entities. Let us assume the KB knows only three common subgraph expressions ρ_1, ρ_2, and ρ_3 for the entities *Rennes* and *Nantes*, such that $\hat{C}(\rho_1) \leq \hat{C}(\rho_2) \leq \hat{C}(\rho_3)$ as illustrated in Figure 1. Each node in the tree is an expression, i.e., a conjunction of subgraph expressions and its complexity $\hat{C}$ is in parentheses. When visiting a node, DFS must test whether the corresponding expression

is an RE, i.e., whether the expression describes exclusively the target entities. If the test fails, the strategy should move to the node's first child. If the test succeeds, DFS must verify whether the expression is less complex than the least complex RE seen so far. If it is the case, this RE should be remembered, and DFS can prune the search space by backtracking. To see why, imagine that $\rho_1 \wedge \rho_2$ in Figure 1 is an RE. In this case, all REs prefixed with this expression (the node's descendants) will also be REs. However, all these REs are more complex. This means that we can stop descending in the tree and prune the node $\rho_1 \wedge \rho_2 \wedge \rho_3$ in Figure 1. We call this step a *pruning by depth*. We can do further pruning if we leverage the order of the entities. In our example, if $\rho_1 \wedge \rho_2$ is an RE, any expression prefixed with $\rho_1 \wedge \rho_i$ for $i > 2$ must be more complex and can be therefore skipped. We call this a *side pruning*. All these ideas are formalized by Algorithm 1 that takes as input a KB $\mathcal{K}$ as well as the entities to describe, and returns an RE of minimal complexity according to $\hat{C}$. For each of the target entities, line 1 calculates (in a BFS fashion) its matching subgraph expressions, and takes those common to all the target entities. The expressions are then sorted by increasing complexity in a priority queue (line 2), which is processed as follows: At each iteration, the least complex subgraph expression ρ is dequeued (line 5) and sent to the subroutine *DFS-REMI* (line 6) with the rest of the queue. This subroutine explores the subtree rooted at ρ and returns the most intuitive RE e' prefixed with ρ. If e' is less complex than the best solution found so far (line 7), we remember it[2]. If *DFS-REMI* returns an empty expression, we can conclude that there is no RE for the target entities T (line 8). To see why, recall that DFS will, in the worst case, combine ρ with all remaining expressions ρ' that are more complex. If none of such combinations is an RE, there is no solution for T in $\mathcal{K}$.

Algorithm 1: REMI

Input: a KB: $\mathcal{K}$, the target entities: T
Output: an RE of minimal complexity: e

1 $G := \bigcap_{t \in T}$ *subgraphs-expressions*(t)
2 create priority queue from G in ascending order by $\hat{C}$
3 $e := \top$
4 **while** $|G| > 0$ **do**
5 $\quad \rho := G.dequeue()$
6 $\quad e' :=$*DFS-REMI*$(\rho, G, T, \mathcal{K})$
7 $\quad$ **if** $\hat{C}(e') < \hat{C}(e)$ **then** $e := e'$
8 $\quad$ **if** $e = \top$ **then return** $\top$
9 **return** e

We implemented Algorithm 1 in Java 8, including a parallel version called P-REMI (detailed in our technical report [3]).

4 EXPERIMENTAL EVALUATION

We evaluated REMI along two dimensions: output quality, and runtime. The evaluation was conducted on two popular KBs, namely DBpedia and Wikidata[3]. Our technical report [3] offers details about the experimental datasets, as well as a more extensive qualitative evaluation of REMI.

[2]We define $\hat{C}(\top) = \infty$
[3]http://dbpedia.org, http://wikidata.org

metric	#participants	p@1	p@2	p@3
$\hat{C}_{fr}$	44	0.38±0.42	0.66±0.18	0.88±0.09
$\hat{C}_{pr}$	48	0.43±0.42	0.53±0.25	0.72±0.16

Table 2: Average precision@k and standard deviation for $\hat{C}$'s ranking of subgraph expressions in DBpedia

4.1 Qualitative Evaluation

We carried out three user studies in order to evaluate REMI's descriptions on instances of the classes person, settlement, album, film, and organization. The cohort consisted mainly of computer science students, researchers, and university staff. It also included some of their friends and family members.

4.1.1 Evaluation of $\hat{C}$. Subgraph expressions are the building blocks of REs, thus intuitive REs should make use of concise and informative pieces. We measure to which extent the function $\hat{C}$ captures intuitiveness by asking the participants to rank a set of 5 subgraph expressions by simplicity and comparing this ranking with the ranking provided by $\hat{C}$. The expressions come from the common subgraph expressions ranked by Alg. 1 (line 2) using $\hat{C}$, and include the top 3 as well as a baseline defined by (i) the worst ranked, and (ii) a random subgraph expression. We manually translated the subgraph expressions to natural language statements in the shortest possible way using the textual descriptions (predicate *rdfs:label*) of the concepts when available. We show the results of our findings on 24 sets of entities in Table 2 for our two variants of $\hat{C}$. We observe that precision@1 is low. This happens because people usually deem the predicate type the simplest whereas REMI often ranks it second or third (16 times for $\hat{C}_{fr}$). This shows the need of special treatment for the *type* predicate as suggested by [6]. Nevertheless, the high values for the other metrics show a positive correlation between the preferences of the users and the function $\hat{C}$. In 88% of the cases, the three simplest subgraph expressions according to $\hat{C}$ are among the three simplest ones according to users.

4.1.2 Evaluation of REMI's output. A second study requested users to rank by simplicity the answer of REMI and a baseline consisting of 2 to 4 additional REs (solutions encountered during search space traversal). The entities were hand-picked to guarantee the existence of at least two REs sufficiently different from each other. Based on our previous findings, we used *fr* as notion of prominence. We report an average MAP (mean average precision) of 0.64±0.17 for this task on 20 sets of entities with 51 answers each, if we assume REMI's solution as the only relevant answer. We recall that a MAP of 1 denotes full agreement between REMI and the users, while a MAP of 0.5 means that REMI's solution is always among the user's top 2 answers.

4.1.3 User's perceived quality. In order to measure the perceived quality of the reported REs, we requested 86 participants to grade the *interestingness* of 35 Wikidata REs in a scale from 1 to 5, where 5 means the user deems the description interesting based on her personal judgment. Our results exhibit an average score of 2.65±0.71, with 11 descriptions scoring at least 3. During the exchanges with the participants, some of them made explicit their preference for short but at the same time informative REs. The latter dimension is related to the notions of pertinence of concepts and narrative interest. For instance, when asked to select between the REs $country(x, N.\ Zealand) \wedge actor(x, C.Lee)$ and $country(x, N.\ Zealand) \wedge actor(x, y) \wedge religion(x, Buddhism)$ for two movies, 95% of the users preferred the first one. Both REs

Language	DBpedia					Wikidata				
	#solutions	amie+	remi	p-remi	speed-up	#sol.	amie+	remi	p-remi	speed-up
Standard	63	97.4k[8]	10.3k[1]	576	13.5kx, 2.44x	44	115.5k[15]	1.06k	76.2	142kx, 4.7x
REMI's	65	508.2k[68]	66.5k[8]	28.9k	5218x, 21.4x	44	608.3k[60]	21.7k	33.8k	6476x, 7.1x

Table 3: REMI's runtime (in seconds) on DBpedia and Wikidata. Speed-ups are provided for P-REMI w.r.t. AMIE and REMI.

had more or less the same length when translated to natural language, but the second one conveys less information and resorts to a domain-unrelated entity (i.e., Buddhism). These observations suggest that prominence captures the notion of simplicity, but it does not always accurately model the dimension of informativeness. While these examples might discourage the use of existential variables in descriptions, we remark that users also liked REs such as $in(x, Brittany) \wedge mayor(x, y) \wedge party(y, Socialist)$ (DBpedia) for Rennes and Nantes, or $actor(x, y) \wedge leader(y, Pisa)$ for the Italian movie "Altri templi" (Wikidata), as they deemed the first one quite pertinent, and the second one narratively interesting. Other interesting REs from DBpedia include "she died of aplastic anemia" for Marie Curie, and "they were both places of the Inca Civil War" for Ecuador and Peru. Finally, we highlight the impact of noise and incompleteness in the quality of the solutions. For instance, REMI cannot describe France as the country with capital Paris, because Paris is also the capital of the former Kingdom of France in DBpedia.

4.2 Runtime Evaluation

4.2.1 Opponent. RE mining can be conceptually formulated as a rule mining task. Hence, we compare the runtime of REMI and a state-of-the-art rule miner designed for large KBs, namely AMIE+ [2]. Given thresholds on support and confidence, AMIE+ mines Horn rules of the form $p(X, Y) \Leftarrow \bigwedge_{1 \leq i \leq n} p_i(X_i, Y_i)$, such as $speaks(x, English) \Leftarrow livesIn(x, UK)$, on RDF KBs . The *support* of a rule is the number of facts correctly predicted by the rule. If we normalize this measure by the total number of predictions made by the rule, we obtain its *confidence*. RE mining for a target entity set T is equivalent to rule mining with AMIE+, if we instruct the system to find rules of the form $\psi(x, True) \Leftarrow \bigwedge_{1 \leq i \leq n} p_i(X_i, Y_i)$, where ψ is a surrogate predicate with facts $\psi(t, True)$ for all $t \in T$. In this case, the right-hand side of the rule becomes our RE. We set thresholds of $|T|$ and 1.0 for support and confidence respectively. This is because an RE should predict the exact set of target entities, neither subsets nor supersets. AMIE+ does not define a complexity score for rules and outputs all REs for the target entities, thus we use $\hat{C}_{fr}$ to rank AMIE's output and return the least complex RE.

4.2.2 Results. We compared the runtimes of REMI and AMIE+ on a server with 48 cores (Intel Xeon E2650 v4), 192GB of RAM[4], and 1.2T of disk space (10K SAS). We tested the systems on 100 sets of DBpedia and Wikidata entities taken from the same classes used in the qualitative evaluation. Small sets of entities are challenging in our setting, so we picked random sets of 1, 2, and 3 entities of the same class in proportions of 50%, 30%, and 20%. We mined REs for those sets of entities according to (i) the standard language bias of conjunctions of bounded atoms, and (ii) REMI's language of conjunctions of subgraph expressions. We show the total runtime among all sets for AMIE+ and REMI in Table 3. The values in red account for the number of timeouts (for a limit of 2 hours), thus cells with red superscripts define runtime lower bounds. We observe that AMIE+ already timed out 23 times with the state-of-the-art language. In particular, AMIE+ is optimized for rules without constant arguments in atoms, such as $livesIn(x, y) \Leftarrow citizenOf(x, y)$, thus its performance is heavily affected when bound variables are allowed in atoms. In contrast REMI and P-REMI are on average 3 and 4 orders of magnitude (up to 142k times) faster than AMIE+ in this language. In the worst case REMI was confronted with a space of 62 subgraph expressions for the state-of-the-art language bias. For REMI's language bias, however, this number increased to 25.2k, which is challenging even for REMI (8 timeouts in total). Despite this boost in complexity, multithreading makes it manageable: P-REMI can be *at least* 4.7x on average faster than REMI for the extended language bias and *at least* 21x faster for the state-of-the-art language, allowing for real-time RE mining. Finally, we observe that the extended language bias slightly increases the chances of finding a solution (column #solutions in Table 3) in DBpedia. This phenomenon is more common among sets with more than one entity.

[4] AMIE assumes the entire KB fits to main memory

5 CONCLUSION AND FUTURE WORK

In this work we have presented REMI, a method to mine intuitive referring expressions on large RDF KBs. REMI builds upon the observation that users prefer prominent entities in descriptions and leverages this fact to quantify the intuitiveness of descriptions in bits. Our results show that (1) real-time RE generation is possible in large KBs and (2) a KB-based frequency ranking can provide intuitive descriptions despite the noise in KBs. This latter factor impedes the fully automatic generation of intuitive REs for NLG purposes, however our descriptions are applicable to scenarios such as computer-aided journalism and query generation. As future work we aim to investigate if external sources—such as search engines or external localized corpora—can yield even more intuitive REs that model users' background more accurately. We also envision to relax the unambiguity constraint to mine REs with exceptions. We provide the source code of REMI as well as the experimental data at https://gitlab.inria.fr/lgalarra/remi.

REFERENCES

[1] Robert Dale. 1992. *Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes.* A Bradford Book, MIT.

[2] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2015. Fast Rule Mining in Ontological Knowledge Bases with AMIE+. *The VLDB Journal* 24, 6 (2015), 707–730.

[3] Luis Galárraga, Julien Delaunay, and Jean-Louis Dessalles. 2019. REMI: Mining Intuitive Referring Expressions on Knowledge Bases. arXiv:cs.AI/1911.01157

[4] Helmut Horacek. 2003. A Best-first Search Algorithm for Generating Referring Expressions. In *Conference on European Chapter of the Association for Computational Linguistics (EACL).*

[5] Emiel Krahmer, Sebastiaan van Erk, and André Verleg. 2003. Graph-based Generation of Referring Expressions. *Computational Linguistics* 29, 1 (2003), 53–72.

[6] Ehud Reiter and Robert Dale. 1992. A Fast Algorithm for the Generation of Referring Expressions. In *Conference on Computational Linguistics (COLING).*

[7] Andreas Thalhammer, Nelia Lasierra, and Achim Rettinger. 2016. LinkSUM: Using Link Analysis to Summarize Entity Data. In *International Conference on Web Engineering (ICWE).*

[8] Arnold Zellner, Hugo A. Keuzenkamp, Michael McAleer, et al. 2001. *Simplicity, Inference and Modelling: Keeping it Sophisticatedly Simple.* Cambridge Univ. Press.